Efficient Mobile Edge Computing for Mobile Internet of Thing in 5G Networks

Yi Zhu
Department Of CSE
Hawaii Pacific University
yzhu@hpu.edu

Kevin Chevalier
Department of CSE
Hawaii Pacific University
kchevali@my.hpu.edu

Xi Wang
Fujitsu Laboratory of America
xi.wang@us.fujitsu.com

Nannan Wang
Fujitsu Laboratory of America
nannan.wang@us.fujitsu.com

Abstract

We study the off-line efficient mobile edge computing (EMEC) problem for a joint computing to process a task both locally and remotely with the objective of minimizing the finishing time. When computing remotely, the time will include the communication and computing time. We first describe the time model, formulate EMEC, prove NP-completeness of EMEC, and show the lower bound. We then provide an integer linear programming (ILP) based algorithm to achieve the optimal solution and give results for small-scale cases. A fully polynomial-time approximation scheme (FPTAS), named Approximation Partition (AP), is provided through converting ILP to the subset sum problem. Numerical results show that both the total data length and the movement have great impact on the time for mobile edge computing. Numerical results also demonstrate that our AP algorithm obtain the finishing time, which is close to the optimal solution.

1. Introduction

In recent years, a significant growth of network connected devices such as smart phones, tablets, laptops, automobiles and drones have been witnessed from the development of *Internet of Thing* (IoT) [1]. This growth results in the use of tens of billions of devices, which require a massive quantity of computational processing and storage resources for sensory data of the environment in autonomous driving, graphics rendering for online gaming, video streaming and more [2]. Across all these applications, data is considered as one of the most significant resources in many applied personal, industrial or academic settings [3]. Furthermore, with the rise of mobile IoT, the availability of large data has increased to meet the requirements of mobile devices to obtain as much information as possible from users and their surroundings to increase the scope of its computing capabilities and maximize success and accuracy for a given task [3].

An efficient solution to the rising computational requirements is *Mobile Edge Computing* (MEC) [2]. This allows mobile devices to offload computing tasks to an edge server, which contains higher processing abilities to decrease the computing time. On the other hand, in comparison to cloud computing, MEC requires edge servers with relatively moderate computational resources but closer to IoT than the cloud to reduce the communication latency [4]. This latency can be further reduced in 5G networks.

In our work, our objective is to design a joint system for mobile edge computing between local execution and computation offloading to minimize the finishing time of a computing task for a mobile device. We name our problem Efficient Mobile Edge Computing (EMEC). In this paper, we make the following assumptions: 1) devices are connected to an edge server through an edge gateway in 5G networks; 2) devices may send the data to an edge gateway and receive the results from the same or different gateway; 3) we consider the dynamic case. For simplicity, we only consider a single mobile device interacting with one or two gateways; 4) the computing task generated by the mobile device consists a set of independent data blocks that have the option to be executed locally or offloaded to the remote edge; 5) the storages and energy at the device, the gateway, and the edge server are large enough to process the communication and computing; 6) no queuing delay is considered at both the mobile device and the edge; and 7) the mobile device is traveling in a fixed direction with a constant velocity.

The rest of the paper is organized as follows. Section 2 discusses the related work to EMEC. Section 3 analyzes the components of the time model, provides the EMEC problem formulation, proves the problem to be NP-complete, and addresses the lower bound of EMEC. The ILP based algorithm to achieve the optimal solution is discussed in Section 4. In Section 5,

URI: https://hdl.handle.net/10125/64514
978-0-9981331-3-3

the approximation algorithm is provided and proved to be a fully polynomial-time approximation scheme (FPTAS). The numerical results are discussed in Section 6. Finally, we conclude our paper and provide future works in Section 7.

2. Related works

The previous work of the EMEC problem can be divided into four categories. First of all, many previous works focused on minimizing the energy consumption with the assumption that devices equipped with a battery have limited power [2, 4-8]. Authors in [4] proposed the evolutionary algorithm to minimize the energy consumption for data offloading. In [2], Yan et al. considered an energy efficient offloading method under unstable channels. The energy consumptions for frequency configuration, transmission power allocation, channel rate scheduling, and offloading strategies were considered in [5-6]. The energy consumption per bit was used in [7] to indicate the energy efficiency of the computing. The objective in [7] was to build the most energy efficient computing system for all users. Our work differs in two major aspects: 1) instead of considering the energy consumption, we consider to minimize the finish time for the computing which is related to the quality of service to users; and 2) instead of just considering the computing and communication, we also consider the mobility of devices.

The second category of previous works studied the mobile data offloading schemes with mobility predictions of devices [8-11]. In those papers, a complex human behavior was grasped through the patterns and a tail matching subsequence algorithm was developed to predict the mobility of devices. After predicting the mobility behavior, a genetic based mobility aware offloading algorithm was proposed to solve single-job, multicomponent, and multisite offloading scenarios. Although our work also considers a single job, multi-component and at most 2-site offloading, our work differs in three major aspects: 1) instead of predicting the mobility behavior, we assume a pre-determined routes with a constant velocity, for example, moving drones and self driven cars; 2) we consider a complicated time model instead of considering a complicated mobile pattern; and 3) instead of heuristic algorithms, we provide a FPTAS to solve the problem approximately.

The third category of previous works focused on data offloading for high-speed vehicles [12-13]. In [12], task-scheduling algorithms were applied to high velocity automobiles to take advantage of the mobility and the processing power of smart cars. The authors in [13] offered offloading mechanisms with the assumption that vehicles have similar computing requests. Although vehicles are among mobile IoT studied in our paper, our work differs in three aspects: 1) instead of taking advantage of processing power of smart cars, we use an edge server to speed up the computing; 2) instead of assuming the similar task processing simultaneously, we consider a single task at one time; and 3) instead of high velocity, we capture the velocity as the moving time between edge gateways, which will affect the offloading capability.

The last category of previous works studied the data offloading through machine learning [14-15]. In [14], Min et al. provided a reinforcement learning based algorithm for energy harvesting devices. The deep learning model learned the optimal offloading policy in respect to the device's internal conditions, transmission conditions, and the energy input. The authors in [15] proposed a Deep-Q-Network based resource allocation algorithm to manage an execution-offloading schedule for multiple users and devices with the objective of minimizing energy consumption and delay. In [15], a single computing task divided the total data into predetermined data blocks that had the option to be offloaded or executed locally. Our work differs in two major aspects: 1) instead of using deep learning, we propose a FPTAS which solves the problem approximately within limited time; and 2) instead of considering energy and delay, our objective focuses on the finishing time which includes computing, communication, and moving time.

3. Time model and problem formulation

In this section, we will first discuss the time needed for when the mobile device to offload the data and obtain the results from the edge. In detail, the model contains three major aspects of time: computing time, communication time, and moving time. We then provide the problem statement of efficient edge computing with the objective of minimizing the finishing time and prove the overall problem is NP-complete. Finally, we show the lower bound of the delay and provide the steps to obtain the solution.

3.1. Time model

When the device offloads data to the gateway, it will suffer communication time. Furthermore, the computing requires time at both the local device and the remote edge server. As for the movement, it limits the time and data amount of communication between the edge and the device.

3.1.1. Computing time. Usually, the computation depends on the input data size and the algorithm. For simplicity, the computing time for the data size L can be approximately calculated as follows:

$$t_1 = \frac{L^\alpha}{f}, \quad (1)$$

where f is the computing frequency, and α is the time complexity index of the algorithm. In specific, the computing time at the device (t_1^D) and the edge (t_1^E) can be calculated as

$$t_1^D = \frac{L_D^\alpha}{f_D}, \quad (2)$$

$$t_1^E = \frac{L_E^\alpha}{f_E}, \quad (3)$$

where L_D and L_E are the amount of data processed at the device and the edge, respectively; and f_D and f_E are the computing frequency at the device and the edge, respectively.

3.1.2. Moving time. Since the device is moving from a to b at the constant speed s, the moving time is

$$t_2 = \frac{|x_b - x_a|}{s}, \quad (4)$$

where x_a and x_b are locations of points a and b, respectively and therefore, $|x_b - x_a|$ represents the distance between a and b.

3.1.3. Communication time. When the device sends the data to the edge in 5G wireless networks, the maximum data rate R (according to Shannon theorem and the channel characteristics of 5G networks in [16]) can be determined by

$$R = \begin{cases} B\log_2(1 + 10^{11.6 - n\log_{10}\frac{4\pi|x|}{\lambda}}), & if\ |x| \le C \\ 0, & otherwise \end{cases},$$

where B is the bandwidth of the channel, n is the power decreasing index of communication environment, λ is the wavelength, C is the communication range of the edge gateway, and $|x|$ is the distance between the device and the edge gateway. Note that x can be negative when the device is at the left side of the edge gateway. If the edge gateway is located at x_c, L_E amount of data is required to be transmitted, and the data transmission starts at a with location x_a ($x_c - C \le x_a \le x_c + C$), the ending point b with location x_b can be obtained through the following equation

$$L_E = \int_{x_a}^{x_b} B\log_2\left(1 + 10^{11.6 - n\log_{10}\frac{4\pi(|x_c - x|)}{\lambda}}\right)dx. \quad (5)$$

Note that $x_b \le x_c + C$ which requires

$$L_E \le \int_{x_a}^{x_c + C} B\log_2\left(1 + 10^{11.6 - n\log_{10}\frac{4\pi(|x_c - x|)}{\lambda}}\right)dx, \quad (6)$$

when we obtain x_b, the communication delay can be indirectly obtained through Eq. (4).

3.1.4. Time for mobile edge computing. Usually, the computing frequency at the edge is more powerful than that at the device. Therefore, the edge computing can fully take advantage of computing capability at the edge.

When the data block with length l is offloaded to compute at the edge, the total time for processing the data block remotely includes the communication time for delivering l amount of data, the computing time at the edge, and another communication time for delivering the results with length βl. Note that $\beta > 0$ is the output ratio between output and input data length.

When the length l and βl are small, the device can offload the data and obtain the results from the same edge gateway. In this case, the movement will not affect the finishing time but just setting the maximum value of l and βl. On the other hand, the device may need a gateway to offload the data and move to the other gateway to obtain the results. In this case, the finishing time can be determined by adding the moving time and communication time of length βl while the communication time of length l and computing time at the edge will determine the location of the start point for delivering the results.

3.2. Problem formulation and NP-completeness

The efficient mobile edge computing (EMEC) problem can be formally described as follows.

Given: a computing task of N data blocks $\mathbf{Y} = \{y_1, y_2, \dots, y_n, \dots y_N\}$ with length $\{l_1, l_2, \dots l_n, \dots, l_N\}$, the location of two edge gateways (x_{c1} and x_{c2}), the time complexity index α and output ratio β, the device computing frequency f_D and the edge computing frequency f_E, the moving speed s, and the time model in 3.1.

Find: a subset $\mathbf{Y}' = \{y'_1, y'_2, \dots, y'_{N'}\}$ of $N' \le N$ data blocks to be processed at the edge and the number of gateways to use.

Objective: the finishing time based on Eqs. (2-5) is minimized.

Constraints:

1) Data partition constraint: each data block should be processed at either edge or device.
2) Communication constraint: the total amount of data offloaded to the edge and the total amount of received results should not exceed the limitation setting in Eq. (6).

Theorem 1. EMEC is NP-complete.

Proof. The decision form of EMEC can be described as follows:

Given: a computing task $\mathbf{Y} = \{y_1, y_2, \dots, y_n, \dots y_N\}$ with length $\{l_1, l_2, \dots l_n, \dots, l_N\}$, the locations of two edge gateways (x_{c1} and x_{c2}), α, β, f_D, f_E, s, the time model in 3.1, and the constant K.

Question: could we find a subset $\mathbf{Y}' = \{y'_1, y'_2, \dots, y'_{N'}\}$ of data blocks and the number of

gateways to use such that the finishing time is no more than *K*?

We can guess the set $\mathbf{Y}'$ within polynomial time. We can calculate the finishing time for $\mathbf{Y}-\mathbf{Y}'$ based on Eq. (2) and calculate the finishing time for $\mathbf{Y}'$ based on Eqs. (3-5). We can also determine the number of gateways based on Eq. (6). All calculations and the answer to the question can be done in polynomial time. Therefore, the EMEC problem belongs to NP class.

We prove EMEC is NP-complete through showing the well-known subset sum (SS) problem is polynomial time reducible to EMEC (i.e. SS$\leq_p$EMEC).

The decision form of SS can be stated as follows:

Given: a set $\mathbf{Z} = \{z_1, z_2, \dots, z_m, \dots z_M\}$ with weight $\{w_1, w_2, \dots w_n, \dots, w_M\}$ and the constant *W*.

Question: could we find a subset $\mathbf{Z}' = \{z'_1, z'_2, \dots, z'_{M'}\}$ with weight $\{w'_1, w'_2, \dots, w'_{M'}\}$ such as $\sum_{i=1}^{M'} w'_i = W$?

Given the instance of SS, we can construct the instance of EMEC as follows:

Let the moving speed $s = 0$, the device is located at one of the edge gateway. By doing so, the data transmission is constant *B'*. Let $\alpha = \beta = 1$, $f_D = \frac{1}{1+\frac{2}{B'}}$ and $f_E = 1$. Let $\mathbf{Y} = \mathbf{Z} \cup \{z_{M+1}\}$ where $N = M + 1$ and $l_{M+1} = (\sum_{i=1}^{M+1} w_i - 2W)$. Let $K = (\sum_{i=1}^{M+1} w_i - W) \cdot (1 + \frac{2}{B'})$.

When the answer to the instance of SS is yes, we have $\mathbf{Z}' = \{z'_1, z'_2, \dots, z'_{M'}\}$ with weight $\{w'_1, w'_2, \dots, w'_{M'}\}$ such as $\sum_{i=1}^{M'} w'_i = W$. Let $\mathbf{Y}' = \mathbf{Z}' \cup \{z_{M+1}\}$ and the total data to be delivered to the edge is $(\sum_{i=1}^{M+1} w_i - W)$ and the delay is $\frac{(\sum_{i=1}^{M+1} w_i - W)}{B'} + \frac{(\sum_{i=1}^{M+1} w_i - W)}{f_E} + \frac{(\sum_{i=1}^{M+1} w_i - W)}{B'} = (\sum_{i=1}^{M+1} w_i - W) \cdot \left(1 + \frac{2}{B'}\right) = K$. The remaining data processed at the device will also be $(\sum_{i=1}^{M+1} w_i - W)$ which requires $\frac{(\sum_{i=1}^{M+1} w_i - W)}{f_D} = (\sum_{i=1}^{M+1} w_i - W) \cdot \left(1 + \frac{2}{B'}\right) = K$. Therefore, EMEC will also answer yes.

On the other hand, when the answer to the instance of EMEC is yes, the length at both sides cannot exceed $(\sum_{i=1}^{M+1} w_i - W)$ while the total data length of $\mathbf{Y}$ is $2(\sum_{i=1}^{M+1} w_i - W)$. Therefore, the length of each side equals to $(\sum_{i=1}^{M+1} w_i - W)$. Now we find the set that contains the element with length $(\sum_{i=1}^{M+1} w_i - 2W)$ and remove that element from the set. By doing so, we find $\mathbf{Z}'$ with the total weight *W*. Therefore, SS will also answer yes.

Based on the above discussion, our problem is also NP-complete. ■

3.3. Lower bound of the finishing time

From the proof of Theorem 1, the lower bound of the finishing time will be achieved when the time $(t_D = t_1^D)$ for processing the data (L_D) at the device equals to the time (t_E) processing the task at the edge remotely, which includes the time for processing the data (L_E) at the edge and the time for delivering the data to/from the edge. In detail, let $L = \sum_{i=1}^{N} l_i$ which can be partitioned into any length between 0 and *L*. The lower bound of the finishing time is shown as follows.

Theorem 2. The lower bound of finishing time for EMEC is t_D where either L_E or βL_E makes the equality in Eq. (6), otherwise $t = t_D = t_E = \frac{L_D^{\alpha}}{f_D}$.

Proof. As for the first part, it is obvious that L_E is the maximum amount of data we can offload due to the communication constraint and $t = t_D > t_E$.

As for the second part of the theorem, we prove it through contradiction. Assume we have the partition (L'_D, L'_E) such that $t'_D < t$ and $t'_E < t$. Since $t'_D < t$, we have $L'_D < L_D$. Thus, we obtain $L'_E = L - L'_D > L - L_D = L_E$ which leads to $t'_E > t_E = t$. This contradicts to our assumption. Therefore we prove the second part.

Based on the above analysis, we obtain the lower bound of the finishing time for EMEC. ■

Through Theorem 2, the lower bound can be calculated through Eqs. (2-6). Furthermore, Eqs. (5-6) can be obtained through bisection search. Although we obtain the best length partition (L_D, L_E), the lower bound may not be achieved since the data block cannot be partitioned. Through the proof of Theorem 2, the optimal solution will be achieved when the total data length (L'_E) offloaded to the edge is closest to L_E. Note that the closest contains two cases: one is $L'_E > L_E$ and the other is $L'_E \leq L_E$. Since the processing time at the edge and the device is different, the optimal solution to EMEC is achieved through the following proposition.

Proposition 3. Given the best length partition (L_D, L_E). The minimum time t_{opt} equals to either the time t'_D for the set of blocks with the total length $L'_D = \min_{L" \geq L_D}(L" - L_D)$ or the time t'_E for the set of blocks with the total length $L'_E = \min_{L_E < L" \leq M_E}(L" - L_E)$ where M_E is the maximum amount of data we can offload based on Eq. (6).

4. ILP based algorithm

In this section, we will develop two ILP models according to two scenarios in proposition 3 and provide an ILP based algorithm to obtain the optimal solution.

Input to the ILP model

N	Number of data blocks

L_E	Data length processed at the edge
L_D	Data length processed at the device
M_E	Maximum data length processed at the edge
$\{l_n\}_N$	Array of data block length
The variable of ILP	
$\{z_n\}_N$	0-1 variable. $z_n = 1$ if the nth data block is offloaded to the edge; otherwise $z_n = 0$.

According to the first part of proposition 3, more time is needed at the device and therefore, we call this model device-centric model (DCM). ILP for DCM can be described as follows.

<DCM>

$$min \quad \sum_{n=1}^{N} l_n \cdot (1 - z_n) \tag{7}$$

$$s.\ t. \quad \sum_{n=1}^{N} l_n \cdot (1 - z_n) \geq L_D \tag{8}$$

Since L_D is given beforehand, we only need to minimize the total data length, shown in Eq. (7), processed at the device. While Eq. (8) sets the constraint that the total length should be greater than or equal to L_D according to proposition 3.

According to the second part of proposition 3, more time is needed at the edge and therefore, we call this model edge-centric model (ECM). ILP for ECM can be described as follows.

<ECM>

$$min \quad \sum_{n=1}^{N} l_n \cdot z_n \tag{9}$$

$$s.\ t. \quad \sum_{n=1}^{N} l_n \cdot z_n \geq L_E \tag{10}$$

$$\sum_{n=1}^{N} l_n \cdot z_n \leq M_E \tag{11}$$

Since L_E is given beforehand, we only need to minimize the total data length, shown in Eq. (9), processed at the edge. While Eq. (10) sets the constraint that the total length should be greater than or equal to L_E and Eq. (11) guarantees the communication constraint obtained by Eq. (6).

Now we provide the whole algorithm to obtain an optimal solution. The key idea is to find the set of data blocks, the total size is close to the best length provided through Theorem 2. Therefore, we name our algorithm as Best Partition (BP) algorithm. The BP algorithm can be described as follows.

Algorithm 1. **Best Partition (BP)**

Input: a computing task of N data blocks $\mathbf{Y} = \{y_1, y_2, \dots, y_n, \dots y_N\}$ with positive integer length $\{l_1, l_2, \dots l_n, \dots, l_N\}$, the location of two edge gateways (x_{c1} and x_{c2}), the start location (x_a), the time complexity index α and output ratio β, the device computing frequency f_D and the edge computing frequency f_E, and the moving speed s.

Output: a subset $\mathbf{Y}' = \{y'_1, y'_2, \dots, y'_{N'}\}$ of $N' \leq N$ data blocks to be processed at the edge, the gateway(s) (c_1 or c_2 or both) to use, and the minimum finishing time (t_{opt}).

Begin

Step 0. Choose c_1 and c_2 as the gateways

Step 1. $L = \sum_{i=1}^{N} l_i$ which can be partitioned into any length between 0 and L.

Step 2a. If two gateways are used, calculate M_{E1} for sending data by setting equality in Eq. (6) and calculate M_{E2} for receiving results by setting equality in Eq. (6). Note that when receiving the results, we need to compare the computing time and moving time to determine the starting point (x'_a) for receiving. We can obtain $M_E = \min\{M_{E1}, \frac{M_{E2}}{\beta}\}$.

Step 2b. Otherwise calculate M_{E1} for sending data by setting equality in Eq. (6) and $M_E = \frac{M_{E1}}{1+\beta}$.

Step 3. Obtain the best length partition (L_D, L_E) for L by calculating Eqs. (2-5). If $L_D < \min_{1 \leq n \leq N} l_n$, $\mathbf{Y}' = \mathbf{Y}$ and goto Step 10. Otherwise,

Step 4. Call ILP solver to solve DCM and obtain the value for $\{z_n\}_N$.

Step 5. Form $\mathbf{Y'_{1D}}$ such that $y'_n \in \mathbf{Y'_{1D}}$ if and only if $z_n = 1$, and calculate t'_D with the total length $\sum_{y'_n \notin \mathbf{Y'_{1D}}} l_n$ based on Eq. (2).

Step 6. If $L_E < M_E$, Call ILP solver to solve ECM and obtain the value for $\{z_n\}_N$; otherwise $t'_E = \infty$.

Step 7. If $\{z_n\}_N$ exists, form $\mathbf{Y'_{1E}}$ such that $y'_n \in \mathbf{Y'_{1E}}$ if and only if $z_n = 1$, and calculate t'_E with the total length $\sum_{y'_n \in \mathbf{Y'_{1E}}} l_n$ based on Eqs. (4-5);

Step 8. If $t'_D \leq t'_E$, $t_{1,opt} = t'_D$ and $\mathbf{Y'_1} = \mathbf{Y'_{1D}}$; otherwise $t_{1,opt} = t'_E$ and $\mathbf{Y'_1} = \mathbf{Y'_{1E}}$.

Step 9. If x_a is close to x_{c1} then we choose c_1 only; otherwise we choose c_2 only.

Step 10. Repeat Step 1-8 once with only one gateway and obtain $t_{2,opt}$ and $\mathbf{Y'_2}$.

Step 11. If $t_{1,opt} \leq t_{2,opt}$, $t_{opt} = t_{1,opt}$ and $\mathbf{Y}' = \mathbf{Y'_1}$; otherwise $t_{opt} = t_{2,opt}$ and $\mathbf{Y}' = \mathbf{Y'_2}$.

End

The total time complexity is dominated by solving one or two ILP models. Since we need to determine N number of binary variables, the total time complexity of the algorithm is $O(2^N)$.

5. Approximation algorithm

Although the ILP based algorithm provides the insight of the EMEC problem, the algorithm can only solve the small-scale problem because EMEC is NP-complete. In this section, we will first develop the approximation algorithm to replace DCM and ECM in steps 4 and 6 of BP. We then provide the whole algorithm, named Approximate Partition (AP). Finally, we conclude this section by showing the AP algorithm is a fully polynomial-time approximation scheme (FPTAS).

5.1. Approximation algorithm for DCM

Through the proof of Theorem 1, the fundamental idea to solve EMEC approximately is to convert EMEC to the subset sum problem. Therefore, we rewrite DCM as follows.

$$max \quad \sum_{n=1}^{N} l_n \cdot z_n \tag{12}$$

$$s.\ t. \quad \sum_{n=1}^{N} l_n \cdot z_n \le L_E \tag{13}$$

The approximation algorithm for DCM can be described as follows.

Algorithm 2. **DCM Approximation**

Input: a length array $\{l_1, l_2, \dots l_n, \dots, l_N\}$, L_E, and $\varepsilon' = \frac{\varepsilon}{e\lceil\alpha\rceil L}$ where $L = \sum_{i=1}^{N} l_i$ and e is Euler's number.

Output: $\{z_n\}_N$ where $z_n = 1$ if the n[th] data block is offloaded to the edge; otherwise $z_n = 0$.

Begin

Step 0. $S_0 \leftarrow \{0\}$, $U_0 = \{\emptyset\}$, and $i = 1$.

Step 1. $S_i \leftarrow S_{i-1}$, $U_i \leftarrow U_{i-1}$, and $j = 1$.

Step 2. $s'_j = s_j + l_i$ where $s_j \in S_{i-1}$, and $u'_j = u_j \cup \{i\}$ where $u_j \in U_{i-1}$.

Step 3. If $s'_j \notin S_i$, then append the element s'_j to the set S_i and append the set z'_j to the collection Z_i.

Step 4. $j \leftarrow j + 1$.

Step 5. Repeat Steps 2-4 until $j = |S_{i-1}|$.

Step 6. $S'_i \leftarrow \{0\}$, $U'_i = \{\emptyset\}$, $v \leftarrow s_1 \in S_i$, and $j \leftarrow 2$.

Step 7. If $s_j > v \cdot (1 + \frac{\varepsilon'}{2N})$, append $s_j \in S_i$ to the set S'_i, append the corresponding set $u_j \in U_i$ to the collection U'_i, and $v \leftarrow s_j \in S_i$.

Step 8. $j \leftarrow j + 1$.

Step 9. Repeat Steps 7-8 until $j = |S_i|$.

Step 10. $S_i \leftarrow S'_i$ and $U_i \leftarrow U'_i$.

Step 11. Remove from S_i every element $s_j > L_E$ and remove its corresponding u_j from the collection U_i.

Step 12. $i \leftarrow i + 1$.

Step 13. Repeat Steps 1-12 until $i = N$.

Step 14. Find the largest element s_i in S_N and its corresponding set u_i.

Step 15. For $1 \le n \le N$, if $n \in u_i$, then $z_n = 1$; otherwise $z_n = 0$.

End

The time complexity of Algorithm 2 is dominated by solving the subset sum problem approximately, which requires $O(\frac{N \cdot \log L_E}{\varepsilon'})$ time.

5.2. Approximation algorithm for ECM

Similarly, we can modify ECM as follows.

$$max \quad \sum_{n=1}^{N} l_n \cdot (1 - z_n) \tag{14}$$

$$s.\ t. \quad \sum_{n=1}^{N} l_n \cdot (1 - z_n) \le L_D \tag{15}$$

$$\sum_{n=1}^{N} l_n \cdot z_n \le M_E \tag{16}$$

Eqs. (14-15) are treated as the subset sum problem with target value L_D. However, we need one more step to check Eq. (16) compared to DCM approximation. The approximation algorithm for ECM is shown as follows.

Algorithm 3. **ECM Approximation**

Input: a length array $\{l_1, l_2, \dots l_n, \dots, l_N\}$, L_D, and $\varepsilon' = \frac{\varepsilon}{e\lceil\alpha\rceil L}$.

Output: $\{z_n\}_N$ where $z_n = 1$ if the n[th] data block is offloaded to the edge; otherwise $z_n = 0$. A Boolean variable flag where flag = true if $\{z_n\}_N$ satisfies Eq. (17); otherwise flag = false which means $\{z_n\}_N$ does not exist.

Begin

Step 0-14 process the same steps as Algorithm 2 by just changing L_E to L_D in Step 11.

Step 15. For $1 \le n \le N$, if $n \in u_i$, then $z_n = 0$; otherwise $z_n = 1$.

Step 16. If $\sum_{n=1}^{N} l_n \cdot z_n \le M_E$, flag = true; otherwise flag = false.

End

Similar to Algorithm 2, the time complexity is $O(\frac{N \cdot \log L_D}{\varepsilon'})$.

5.3. Approximation partition algorithm

The whole algorithm is described as follows.

Algorithm 4. **Approximation Partition (AP)**

Input: a computing task of N data blocks $\mathbf{Y} = \{y_1, y_2, \dots, y_n, \dots y_N\}$ with length $\{l_1, l_2, \dots l_n, \dots, l_N\}$, the location of two edge gateways (x_{c1} and x_{c2}), the start location (x_a), α, β, f_D, f_E, s, and ε.

Output: a subset $\mathbf{Y}'_{app} = \{y'_1, y'_2, \dots, y'_{N'}\}$ of $N' \le N$ data blocks to be processed at the edge, the gateway(s) (c_1 or c_2 or both) to use, and the approximate finishing time (t_{app}).

Begin

Step 0. Choose c_1 and c_2 as the gateways

Step 1-3. Process the same steps as Algorithm 1 (BP) to obtain (L_D, L_E) for $L = \sum_{i=1}^{N} l_i$ and M_E.

Step 4. Call DCM approximation (Algorithm 2) to obtain the value for $\{z_n\}_N$.

Step 5. Form $\mathbf{Y'_{1D}}$ such that $y'_n \in \mathbf{Y'_{1D}}$ if and only if $z_n = 1$, and calculate t'_D with the total length $\sum_{y'_n \notin \mathbf{Y'_{1D}}} l_n$ based on Eq. (2).

Step 6. If $L_E < M_E$, Call ECM approximation (Algorithm 3) to obtain the value for $\{z_n\}_N$.

Step 7. If flag = true, form $\mathbf{Y'_{1E}}$ such that $y'_n \in \mathbf{Y'_{1E}}$ if and only if $z_n = 1$, and calculate t'_E with the total length $\sum_{y'_n \in \mathbf{Y'_{1E}}} l_n$ based on Eqs. (4-5); otherwise $t'_E = \infty$.

Step 8. If $t'_D \le t'_E$, $t_{1,app} = t'_D$ and $\mathbf{Y'_{1,app}} = \mathbf{Y'_{1D}}$; otherwise $t_{1,app} = t'_E$ and $\mathbf{Y'_{1,app}} = \mathbf{Y'_{1E}}$.

Step 9. If x_a is close to x_{c1} then we choose c_1 only; otherwise we choose c_2 only.

Step 10. Repeat Step 1-8 once with only one gateway and obtain $t_{2,app}$ and $\mathbf{Y}'_{\mathbf{2},app}$.

Step 11. If $t_{1,app} \le t_{2,app}$, $t_{app} = t_{1,app}$ and $\mathbf{Y}'_{app} = \mathbf{Y}'_{\mathbf{1},app}$; otherwise $t_{app} = t_{2,app}$ and $\mathbf{Y}'_{app} = \mathbf{Y}'_{\mathbf{2},app}$.

End

The time complexity of Algorithm 4 can be determined by DCM and ECM approximation algorithms, which requires $O\left(\frac{N\cdot\log L_D}{\varepsilon'} + \frac{N\cdot\log L_E}{\varepsilon'}\right) = O\left(\frac{N\cdot\log L}{\varepsilon'}\right)$ time since $L \ge \max\{L_D, L_E\}$.

5.4. FPTAS proof

In this subsection, we will prove our AP algorithm is a fully polynomial-time approximation scheme (FPTAS), which requires us to prove two parts:

1) The solution is within a factor of $1+\varepsilon$ of the optimal solution; and
2) The running time is polynomial in both N and $\frac{1}{\varepsilon}$.

We will prove those two parts through the following two theorems.

Theorem 4. For $0 < \varepsilon = e\cdot\lceil\alpha\rceil\cdot L\cdot\varepsilon' < 1$, the approximation algorithm achieves the solution within a factor of $1+\varepsilon$ of the optimal solution, where $L = \sum_{i=1}^{N} l_i$ and e is Euler's number.

Proof. Suppose the optimal solution is t_{opt} and the approximation solution is t_{app}. The best length partition to achieve the lower bound of the finishing time is $(L_D,\ L_E)$. First of all, DCM approximation algorithm can achieve the solution within a factor of $1+\varepsilon'$ of the optimal solution [17]. Therefore, we have

$$\frac{L_{E,opt}}{L_{E,app}} \le 1+\varepsilon', \text{ and} \quad (17)$$

$$\frac{L_{D,app}}{L_{D,opt}} = \frac{L-L_{E,app}}{L_{D,opt}} = \frac{L_{D,opt}+L_{E,opt}-L_{E,app}}{L_{D,opt}}. \quad (18)$$

Based on Eqs. (17-18), we obtain

$$\frac{L_{D,opt}+L_{E,opt}-L_{E,app}}{L_{D,opt}} = \frac{L_{D,opt}}{L_{D,opt}} + \frac{L_{E,opt}-L_{E,app}}{L_{D,opt}}$$

$$\le 1 + \frac{L_{E.opt}-\frac{L_{E,opt}}{1+\varepsilon'}}{L_{D,opt}} = 1 + \frac{L_{E.opt}}{L_{D,opt}}\cdot\frac{\varepsilon'}{1+\varepsilon'}. \quad (19)$$

Since DCM approximation algorithm sets L_E as the target value for the subset sum problem, we have $L_{E.opt} \le L_E$ and $L_{D,opt} = L - L_{E.opt} \ge L - L_E = L_D$. Based on those two inequalities into Eqs. (18-19), we obtain

$$\frac{L_{D,app}}{L_{D,opt}} \le 1 + \frac{L_{E.opt}}{L_{D,opt}}\cdot\frac{\varepsilon'}{1+\varepsilon'} \le 1 + \frac{L_E}{L_D}\cdot\varepsilon'. \quad (20)$$

According to Algorithm 4, the finishing time is the smaller value between the time based on the partition in DCM approximation and the time based on the partition in ECM approximation. Assume the finishing time obtained based on the partition in DCM approximation is t_{DCM}. From Eqs. (2), we have

$$\frac{t_{app}}{t_{opt}} \le \frac{t_{DCM}}{t_{opt}} = \frac{L_{D,app}^{\alpha}/f_D}{L_{D,opt}^{\alpha}/f_D} = \left(\frac{L_{D,app}}{L_{D,opt}}\right)^{\alpha}$$

$$\le \left(1+\frac{L_E}{L_D}\cdot\varepsilon'\right)^{\alpha} \le (1+L\cdot\varepsilon')^{\alpha}. \quad (21)$$

Note that $\alpha > 0$. We discuss the approximation ratio through two cases.

Case 1. $\lceil\alpha\rceil = 1$.

$$\frac{t_{app}}{t_{opt}} \le (1+L\cdot\varepsilon')^{\alpha} \le (1+L\cdot\varepsilon')^{\lceil\alpha\rceil} = 1+L\cdot\varepsilon'. \quad (22)$$

Since $\varepsilon = e\cdot\lceil\alpha\rceil\cdot L\cdot\varepsilon'$, we obtain $L\cdot\varepsilon' = \frac{\varepsilon}{e\cdot\lceil\alpha\rceil} = \frac{\varepsilon}{e} \le \varepsilon$. Therefore,

$$\frac{t_{app}}{t_{opt}} \le 1+L\cdot\varepsilon' \le 1+\varepsilon. \quad (23)$$

Case 2. $\lceil\alpha\rceil \ge 2$.

$$\frac{t_{app}}{t_{opt}} \le (1+L\cdot\varepsilon')^{\alpha} \le (1+L\cdot\varepsilon')^{\lceil\alpha\rceil}$$

$$= 1 + \lceil\alpha\rceil L\varepsilon'\left[1 + \frac{1}{2}\binom{\lceil\alpha\rceil-1}{1}L\varepsilon'\right.$$

$$+ \frac{1}{3}\binom{\lceil\alpha\rceil-1}{2}(L\varepsilon')^2 + \cdots + \frac{1}{\lceil\alpha\rceil-1}\binom{\lceil\alpha\rceil-1}{\lceil\alpha\rceil-2}(L\varepsilon')^{\lceil\alpha\rceil-2}$$

$$\left. + \frac{(L\varepsilon')^{\lceil\alpha\rceil-1}}{\lceil\alpha\rceil}\right]$$

$$\le 1 + \lceil\alpha\rceil L\varepsilon'(1+L\varepsilon')^{\lceil\alpha\rceil-1} \le 1 + \lceil\alpha\rceil L\varepsilon'(1+L\varepsilon')^{\frac{1}{L\varepsilon'}}$$

$$\le 1 + e\lceil\alpha\rceil L\varepsilon' = 1+\varepsilon. \quad (24)$$

Note that $e\cdot\lceil\alpha\rceil\cdot L\cdot\varepsilon' < 1$ which implies $\lceil\alpha\rceil - 1 < \lceil\alpha\rceil < \frac{1}{e\cdot L\cdot\varepsilon'} < \frac{1}{L\cdot\varepsilon'}$. Based on two cases (Eqs. (23-24)), we proved this theorem. ■

Theorem 5. For $0 < \varepsilon = e\cdot\lceil\alpha\rceil\cdot L\cdot\varepsilon' < 1$, the time complexity is polynomial in both N and $\frac{1}{\varepsilon}$.

Proof. Since $\varepsilon = e\cdot\lceil\alpha\rceil\cdot L\cdot\varepsilon'$ and Algorithm 4 requires $O\left(\frac{N\cdot\log L}{\varepsilon'}\right)$, the running time is $O\left(\frac{\lceil\alpha\rceil\cdot N\cdot L\cdot\log L}{\varepsilon}\right) = O\left(\frac{\lceil\alpha\rceil\cdot N^2\cdot\log N}{\varepsilon}\right)$ which is polynomial in both N and $\frac{1}{\varepsilon}$. Note that $L \le N\cdot\max\{l_1, l_2, \ldots l_n, \ldots, l_N\}$. ■

Based on Theorems 4 and 5, we have proved the approximation partition algorithm is a fully polynomial-time approximation scheme (FPTAS).

6. Numerical results

In this section, we present numerical results to show that the AP algorithm achieves good performance in both small-scale and large-scale cases.

For comparison purpose, we collect the optimal solution from the ILP based algorithm (BP) for small-scale cases and we obtain the lower bound of the finishing time for large-scale cases.

6.1. Mobile edge computing in small-scale cases

In this subsection, we show that AP can obtain results close to the optimal solution. The device computing frequency $f_D = 1$ GHz and the edge computing frequency $f_E = 50$ GHz. In 5G networks, we set the bandwidth as 1 GHz, communication range as 200m, power decreasing index as 1.8 (outdoor environment), and wavelength as 5 mm. Both time complexity index (α) and output ratio (β) are set to be 1. The distance between two gateways is 400m and we generate the speed of the device as 20, 60, 100, and 140 m/s. The device is 200m left to the first gateway and moves at the given speed to the right. The total data length L is changed from 5 Gb to 10 Gb. The data length of each data block is uniformly distributed between 5% and 10% of L.

Table 1 gives the results of the ILP-based algorithm (BP) as well as the approximation approach (AP). The BP algorithm is running through CPLEX 12.9. Each value shown in the table is the average among 100 running cases. We also compare the difference between results in Table 1.

From Table 1, we first find out that the finishing time increases when the total data length increases because more time is needed to process the data. Furthermore, when we double the data length, the time is more than doubled because the data will transfer twice (one is sending the data to the gateway and the other is receiving results from gateway). Secondly, when the speed increases, the finishing time will first decrease and then increase again. The reason behind this is that based on the start point of the device (which equals to the communication range of the gateway), the device cannot access strong network conditions when it moves slowly. The lower transmission speed limits the amount of data to be offloaded to the edge, which causes large volume calculation processed at the device. On the other hand, when the device moves fast, it pasts the gateway quickly and also limits the amount of data to be offloaded. Finally, we find that our approximation algorithm (AP) achieves the performance close to the optimal solution (obtained by BP). The largest difference ratio is just 0.13%.

Table 1 Comparison between BP and AP

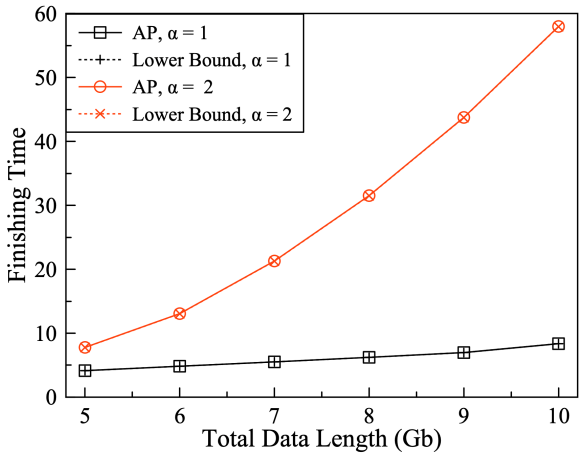
L (Gb)	Speed (m/s)	Finishing Time		Diff. ratio (%)
		BP	AP	
5	20	4.510	4.511	0.02
	60	3.912	3.915	0.08
	100	4.021	4.023	0.05
	140	4.303	4.304	0.02
6	20	5.312	5.319	0.13
	60	4.651	4.655	0.09
	100	5.022	5.024	0.04
	140	5.319	5.320	0.02
7	20	6.090	6.097	0.11
	60	5.435	5.439	0.07
	100	6.025	6.026	0.02
	140	6.318	6.318	0
8	20	6.841	6.844	0.04
	60	6.370	6.373	0.05
	100	7.034	7.035	0.01
	140	7.324	7.324	0
9	20	7.578	7.580	0.03
	60	7.370	7.373	0.04
	100	8.056	8.056	0
	140	8.326	8.326	0
10	20	8.295	8.299	0.05
	60	8.372	8.374	0.02
	100	9.041	9.042	0.01
	140	9.335	9.335	0

6.2. Effect of total data length L on EMEC

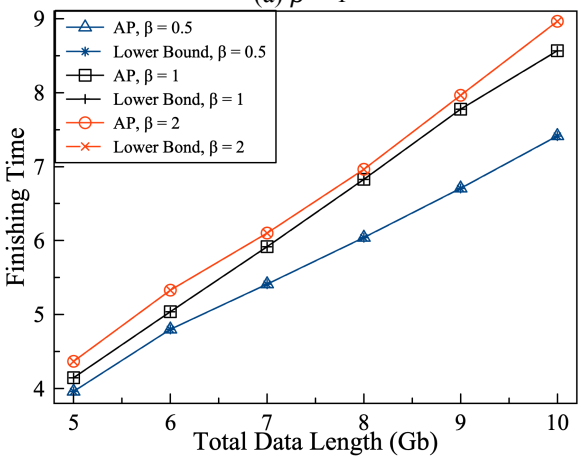
In this experiment, we show how the total data length (L) affects the performance of the mobile edge computing. The computing frequencies and the parameters for 5G communication are the same as Subsection 6.1. The distance between two gateways is 400m. The device is 200m left to the first gateway and moves at the constant speed of 60m/s to the right. We set the time complexity index (α) to be 1 or 2 and the output ratio (β) to be 0.5, 1, or 2. The total data length L is changed from 5 Gb to 10 Gb. The data length of each data block is uniformly distributed between 0.1% and 5% of the total data length L.

Figure 1 shows how the total data length affects the finishing time for a computing task. Each point in Fig. 1 is the average value of 10,000 running cases.

From Fig. 1, we see that the finishing time increases when the total data length increases, which shows the same trend as our small cases. Secondly, we find that when the total data length increases from 5 Gb to 10 Gb, the finishing time is a little more than double (shown as the black line in Fig. 1(a) and in Fig. 1(b)) due to the communication time. However, the finishing time is around 8 times when the time complexity index is 2 (shown as red line in Fig. 1(a)) since more time is required for computing and communication. Thirdly, compared Fig. 1(a) and Fig. 1(b), we find that the time complexity index has more impacts on the finishing time than the output ratio has. The reason behinds this is that the time complexity index is the power index to the data length (see Eq. (2)) but the output ratio is the coefficient to the data length. Therefore, algorithms with low time complexity is more suitable to the mobile edge computing than those requiring much more complicated computation. Finally, we find that our BP algorithm achieves the performance very close



(a) $\beta = 1$



(b) $\alpha = 1$

Fig. 1. Finishing time vs. total data length (L)

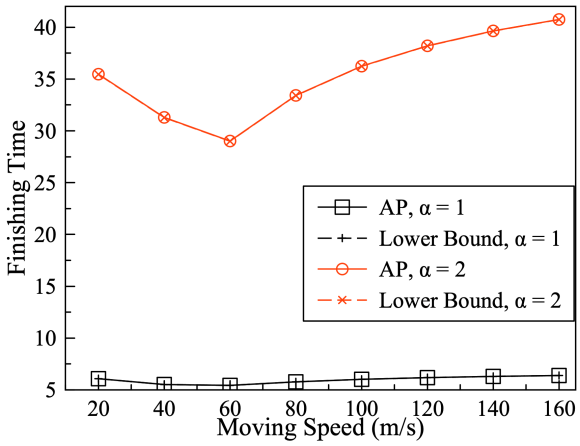
to the lower bound, which demonstrates the effectiveness of FPTAS.

6.3. Effect of the movement on EMEC

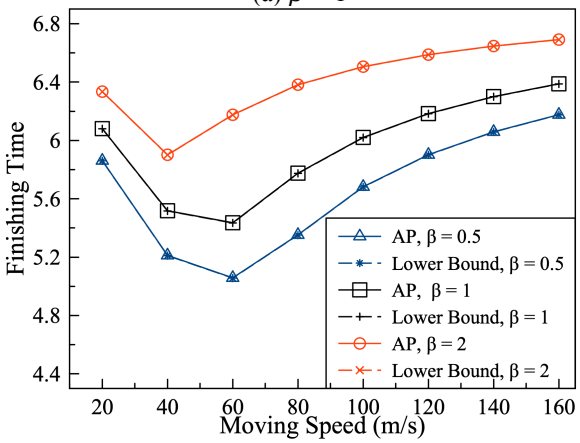
In this experiment, we show how the movement affects the performance of the mobile edge computing. The computing frequencies and the parameters for 5G communication are the same as Subsection 6.1. We set the time complexity index (α) to be 1 or 2 and the output ratio (β) to be 0.5, 1, or 2. The total data length L equals to 7 Gb. The data length of each data block is uniformly distributed between 0.1% and 5% of the total data length L. The distance between two gateways is 400m. The device is 200m left to the first gateway and moves to the right at the speed from 20m/s to 160m/s.

Figure 2 shows how the movement affects the finishing time for a computing task. Each point in Fig. 2 is the average value of 10,000 running cases.

From Fig. 2, we first find that the time decreases and then increases with the same reason discussed in



(a) $\beta = 1$



(b) $\alpha = 1$

Fig. 2. Finishing time vs. the moving speed (s)

subsection 6.1. Based on this finding, there exists the best speed for the mobile edge computing which is the function of total data length and the distance between two gateways. For example, the best speed in our simulation scenario is around 60 m/s. Secondly, we also find that the time complexity index has more impacts on the finishing time than the output ratio when we compare Fig. 2(a) and Fig. 2(b). Finally, our BP algorithm achieves the same trends and almost the same finishing time as the lower bound.

7. Conclusion and future works

In this paper, we study the efficient mobile edge computing (EMEC) problem for a computing task, which can be partitioned into multiple independent data blocks and each of them can be to process either at the device locally or at the edge remotely. When the data block is chosen to process remotely, it will be sent to a fixed located gateway in 5G networks and processed at the edge. The results will be returned back through the same or different gateway. Since the device is moving, this will affect the amount of data to

be offloaded. EMEC determines which blocks will be processed remotely so as to minimize the finishing time. We propose the time model to capture all possible time to process the data. We then formulate the problem, prove that the problem is NP-complete, and provide the lower bound of the finishing time for the given computing task. An ILP-based algorithm (BP) is proposed to obtain the optimal solution. An approximation algorithm (AP), which converts two ILP models in BP to subset-sum problems, is proposed and proved to be a FPTAS. Numerical results demonstrate that AP obtains the finishing time close to the optimal solution in both small-scale and large-scale cases. Numerical results also show that the data length and the movement have great impacts on the finishing time. Compared to the output ratio (β), the time complexity index (α) shows greater impacts on the finishing time.

One possible future work considers the mobile edge computing with dependent data blocks. Another possible future work may consider the data offloading when the edge has the capacity limitation.

8. References

[1] G.R. Alam, M.M. Hassan, Z. Uddin, A. Almogren, and G. Fortino, "Autonomic computation offloading in mobile edge for IoT applications", Future Generation Computer Systems, Elsevier, 2019, pp. 149-157.

[2] H. Yan, X. Zhang, H. Chen, Y. Zhou, W. Bao, and L.T. Yang, "DEED: Dynamic Energy-Efficient Data offloading for IoT applications under unstable channel conditions", Future Generation Computer Systems, Elsevier, 2019, pp. 425-437.

[3] K. Lin, S. Pankaj, and D. Wang, "Task offloading and resource allocation for edge-of-things computing on smart healthcare systems", Computers & Electrical Engineering, Elsevier, 2018, pp. 348-360.

[4] X. Xu, Q. Liu, Y. Luo, K. Peng, X. Zhang, S. Meng, and L. Qui, "A computation offloading method over big data for IoT-enabled cloud-edge computing", Future Generation Computer Systems, Elsevier, 2019, pp. 522-533.

[5] C. Li, J. Tang, and Y. Luo, "Dynamic multi-user computation offloading for wireless powered mobile edge computing", Journal of Network and Computer Applications, Elsevier, 2019, pp. 1-15.

[6] Q. Wang, S. Guo, J. Liu, and Y. Yang, "Energy-efficient computation offloading and resource allocation for delay-sensitive mobile edge computing", Sustainable Computing: Informatics and Systems, Elsevier, 2019, pp. 154-164.

[7] H. Sun, F. Zhou, and R.Q. Hu, "Joint Offloading and Computation Energy Efficiency Maximization in a Mobile Edge Computing System", Transactions on Vehicular Technology, IEEE, 2019, pp. 3052-3056.

[8] V.A. Siris, and K. Dimitrios, "Enhancing mobile data offloading with mobility prediction and prefetching", SIGMOBILE Mobile Computing and Communications Review, ACM, 2013, pp. 22-29.

[9] Y. Shi, S. Chen, and X. Xu, "MAGA: A mobility-aware computation offloading decision for distributed mobile cloud computing", Internet of Things Journal, IEEE, 2017, pp. 164-174.

[10] V.A. Siris, and K. Dimitrios, "Enhancing mobile data offloading with mobility prediction and prefetching", SIGMOBILE Mobile Computing and Communications Review, ACM, 2013, pp. 22-29.

[11] K. Lee, and I. Shin, "User mobility model based computation offloading decision for mobile cloud", Journal of Computing Science and Engineering, KIISE, 2015, pp. 155-162.

[12] H. Dai, X. Zeng, Z. Yu, and T. Wang, "A scheduling algorithm for autonomous driving tasks on mobile edge computing servers", Journal of Systems Architecture, Elsevier, 2019, pp. 14-23.

[13] G. Qiao, S. Leng, K. Zhang, and Y. He, "Collaborative task offloading in vehicular edge multi-access networks", Communications Magazine, IEEE, 2018, pp. 48-54.

[14] M. Min, L. Xiao, Y. Chen, P. Cheng, D. Wu, and W. Zhuang, "Learning-based computation offloading for IoT devices with energy harvesting", Transactions on Vehicular Technology, IEEE, 2019, pp. 1930-1941.

[15] L. Huang, X. Feng, C. Zhang, and L. Qian, Y. Wu, "Deep reinforcement learning-based joint task offloading and bandwidth allocation for multi-user mobile edge computing", Digital Communications and Networks, Elsevier, 2019, pp. 10-17.

[16] J. Wang, H. Zhang, T. Lu, and T.A. Gulliver, "Capacity of 60GHz wireless communication systems over ricean fading channels", Pacific Rim Conference on Communications, Computers.

[17] H. Kellerer, R. Mansini, U. Pferschy, and M. G. Speranza, "An efficient fully polynomial approximation scheme for the subset-sum problem", Jounral of Computer and System Sciences, Elsevier, 2003, pp. 349-370.